Beyond Validation: using programmed diagnostics to learn about, monitor, and successfully complete your DH project

Schema-based validation of XML documents has long been a fundamental tool for quality control in digital edition projects, and the emergence of richer schema languages (RELAX NG and XML Schema) along with adjuncts such as Schematron has greatly improved the constraints and controls available to XML authors and encoders (Jacinto et al. 2002). However, schema-based validation typically takes place at the document level, whereas "most programs that use XML require information that is not encoded in the XML instance or in the schema that governs it" (Vorthmann & Robie 2001). The modern digital edition project typically consists of multiple documents with large numbers of pointers between them: links between named entities and authority lists such as personographies, placeographies and bibliographies;[1] pointers to external documents and fragments, images and other media; and similar complex interrelationships within the collection, and to external resources and authority lists. These relationships need to be tested, checked, and validated too, but it is impractical to do this using document-level schemas. As Durand et al. (2009) point out, "such testing requirements are in fact closer to conventional system or software testing requirements than to document testing in a narrow sense." Most large- and medium-scale projects develop their own methods, programmed and/or impromptu, for addressing these problems, and these have been quite well-described and documented for enterprise-level and corporate contexts,[2] but little has been published on project-level diagnostic testing for XML-based digital edition collections.[3]

In our work as part of Endings[4], an umbrella project that comprises four diverse digital edition projects from different fields, we have been developing a structured approach to implementing methods for checking and enforcing project correctness, consistency, and coherence, which we will describe in this paper. Influenced by Star Trek[5], we have long referred to these processes as "diagnostics", and in our description we follow the franchise tradition detailed in Sternbach and Okuda (1991) in dividing diagnostics into levels; however, we depart from convention in ordering our levels from most granular/least comprehensive up to the most general. For each level, we provide real examples of processes applied to one of our projects.

1 Our use of the terms "personography" and "placeography" follows *The Map of Early Modern London*'s terminology. We use these terms because our resources differ from ordinary bibliographical lists or gazetteers in that both may contain fictional and mythological entities or locations.

2 For instance, see the papers (particularly Waldt 2012) presented at the International Symposium on Quality Assurance and Quality Control in XML, http://www.balisage.net/Proceedings/vol9/contents.html.

3 Rahtz (2007) hints at project-level consistency checking, suggesting that editors do not "rely on the schema" and find alternatives "beyond Schema and DTDs."

4 Endings (https://onlineacademiccommunity.uvic.ca/endingsproject/) is supported by the Social Sciences and Humanities Research Council of Canada.

5 For those unfamiliar with the series, Sternbach & Okuda explain that "all key operating systems and subsystems aboard the *Enterprise* have a number of preprogrammed diagnostic software and procedures… generally classified into five different levels, each offering a different degree of crew verification of automated tests" (46).

We stress that these diagnostics assume that a project is already making full use of RelaxNG and Schematron schemas to for basic constraint. In the case of our projects, we use highly-customized versions of the TEI schema (all TEI conformant[6]) in addition to project-specific Schematron rules, which not only police tagging practices (e.g. enforcing the use of private URI schemes in pointing attributes, and checking the presence of appropriate custom dating attributes for pre-Gregorian dates), but also impose style guide rules such as prohibiting the use of straight apostrophes in document text. We are not suggesting that the diagnostics process described below is meant to replace RelaxNG or Schematron validation, but instead we suggest that, for large scale XML-based projects, diagnostics are crucial for ensuring both document-level and project-level coherence, consistency, and correctness.

Diagnostics

Our diagnostic processes normally take the form of Ant scripts[7] and XSLT transformations, and are run on a Jenkins Continuous Integration server; every time changes are committed to a project repository, the Jenkins server checks out the changes, validates all documents, and runs the entire set of diagnostics processes, providing the results in the form of a public web page such as this one:

6 There has been considerable disagreement about the meaning of “TEI Conformant” over many years (see, for example, Burnard 2017). We use it to mean that any document which validates against our TEI-derived schema will also validate against the complete TEI schema “tei_all”.
7 Ant (http://ant.apache.org/) is a Java tool for managing software build processes.

Statistics ▼

TEI documents found:	1561
<bibl> entries found:	1507
<person> entries found:	3650
<org> entries found:	106
glossary entries found:	84
<ref>s pointing to tagged toponyms found:	16076
<ref>s pointing at bibliographic items (in BIBL1.xml) found:	4504
<ref>s pointing at internal bibliographic items (i.e. mol:bibls) found:	139
<name>s pointing at people (in PERS1.xml) found:	23268
<name>s pointing at organizations (in ORGS1.xml) found:	1065
internal links found:	50347

Consistency Checks

Ill-formed xml:id attributes in pages (0) ▶

Bad internal links in pages (12) ▼

Explanation

- `pantzer.xml` (file:/var/lib/jenkins/jobs/MoEML/workspace/db/data/finding_aids/pantzer.xml)
  - mol:SMOW1
  - mol:THES2
- `PERS1.xml` (file:/var/lib/jenkins/jobs/MoEML/workspace/db/data/PERS1.xml)
  - mol:https://en.wikipedia.org/wiki/Swithwulf_(bishop_of_London)
- `stow_1598.xml` (file:/var/lib/jenkins/jobs/MoEML/workspace/db/data/stow/1598/stow_1598.xml)
  - mol:FILD1
  - mol:DOWN5
  - mol:COL13
  - mol:FLEM6
  - mol:CHAM10
  - mol:WALE2
  - mol:WARF3
  - mol:EKEU2
  - mol:EGGB1

Bad external links (404s) (65) ▶

Redirected links (HTTP 301) (313) ▶

Figure 1: A diagnostics output page from the *Map of Early Modern London* (*MoEML*) project.[8]

In combination with this paper, which is intended to be a useful primer and guide, we have developed a Diagnostics project hosted on GitHub (http://github.com/projectEndings/diagnostics) that can be used by researchers whose digital edition projects have grown to the point where ad hoc manual checking has become impractical. This tool provides a generic set of diagnostic checks that can be applied to any set of TEI XML files. This tool is meant to serve as a foundation for further work. As the specific implementations of diagnostics below demonstrate, individual projects will need to create their own diagnostic checks that address the project's specific encoding methods as well as project specific milestones. The code will probably need to be tailored for each individual project's encoding methods; the Github project provides examples and instructions for adding additional, project-specific diagnostic checks to the base XSLT transformation.

Level 1

Level 1 diagnostics provide project-level, as opposed to document-level, consistency checking to establish the internal coherence of the project, primarily through ensuring referential integrity.[9] This includes checking for pointers to non-existent targets, duplicate @xml:ids within the project, and erroneously encoded references (e.g. tagging a place name as a bibliography reference). Ensuring referential integrity is particularly complex for projects that use "abbreviated pointers" to facilitate internal linking,[10] since it may not be obvious to the encoder or the text editing program which resource is being referenced by a pointer. Thus, the first level of diagnostics checks both whether or not an object pointed to actually exists *and* whether or not the markup correctly represents the relationship between the element and the target resource. For instance, to check all instances of the relationship shown in Fig. 2, a number of different tests are actually done:

8 Although most of the diagnostic processes are carried out with pure XSLT, a few (such as external link checking) require the use of other command-line tools such as LinkChecker (https://wummel.github.io/linkchecker/).

9 We borrow the phrase "referential integrity" from the MLA's "Guiding Questions for Vetters of Scholarly Editions" (2011), which advises peer-reviewers of digital editions that link to multiple databases to see if "referential integrity [is] enforced within the database(s)." The term originates in the relational database field.

10 See TEI Consortium (2016), http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SA.html#SAPU.

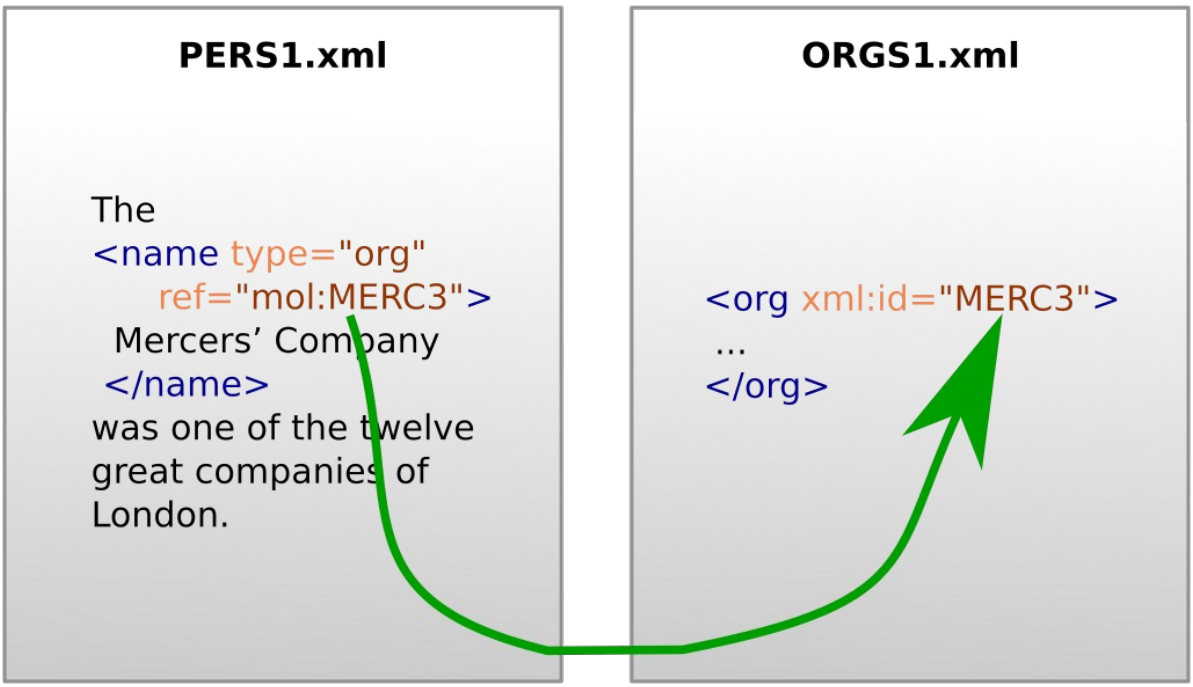


Figure 2: a simple referential integrity check.

1. Every <name type="org"> points at an @xml:id which exists in the project.
2. The element pointed at by <name type="org"> is an <org> element in the ORGS1.xml document.
3. Every <name> element which points at an <org> element in ORGS1.xml has @type="org".

For small-scale projects, this kind of referential integrity check could be accomplished with Schematron, since a Schematron rule using XPath 2.0 can read external documents; such a referential integrity check could be written as follows:

```
<sch:let name="thisDocUri" value="document-uri(/)"/>
<sch:let name="projectUri" value="replace($thisDocUri,'^(.+/data/).+', '$1')"/>
<sch:let name="docCollection"
value="collection(concat($projectUri,'?select=*.xml;&amp;recurse=yes'))"/>
<sch:let name="orgsFile" value="$docCollection//tei:TEI[@xml:id='ORGS1']"/>
<sch:let name="orgIds" value="$orgsFile/descendant::tei:org/@xml:id"/>

<sch:pattern>
 <sch:rule context="tei:name[@ref][starts-with(@ref,'mol:')][@type='org']">
  <sch:let name="ref" value="substring-after(@ref,'mol:')"/>
  <sch:assert test="$ref=$orgIds">
   ERROR: Reference to <sch:value-of select="$ref"/> not found in
   ORGS1.xml.
  </sch:assert>
 </sch:rule>
</sch:pattern>
```

The above schematron pattern loads the external orgography file (ORGS1) and investigates whether or not the value of the @ref attribute corresponds with a declared organization in the orgography. If the pointer token cannot be found within the orgography, then a schematron error is raised.

However, for a project of any significant size, this is impractical. The *Map of Early Modern London* project, for instance, contains at the time of writing 44,167 <name> elements with pointers to <person>s, <org>s, and similar elements. When added to *MoEML*'s standalone schematron, this rule adds nearly a second to the command-line validation of a relatively simple encyclopedia entry, while adding two seconds to validation time for the more densely encoded and interlinked edition of John Stow's 1598 *Survey of London*. A further check for references to persons adds three more seconds to command-line validation of Stow. Simply checking that a linked location exists requires the processing of over a thousand files in this project, since each location is a distinct file. While the addition of a few seconds may be less of a concern for command-line validation, adding several seconds to validation in the Oxygen XML editor inevitably causes frustration for editors working in the files.

Level 2

While Level 1 diagnostics generally focus on coherence and consistency, Level 2 is more concerned with completeness. Level 2 diagnostics monitor progress, generate to-do lists, and identify situations that may indicate error, but require human judgement. These include cases in which:

- Two bibliography or personography entries appear sufficiently similar that they may be duplicates.
- Several <name> elements point to the same authority record, but the text of one of them is significantly different from the others, so it may point at the wrong target.
- A document in the project is not linked from anywhere else, and therefore cannot be "reached".

Such issues cannot be automatically rectified—they are not necessarily errors—but they must be examined. Figure 3 shows an example of the first check, which uses the Universal Similarity Metric (Holmes 2010) to identify potential duplicate bibliography entries.

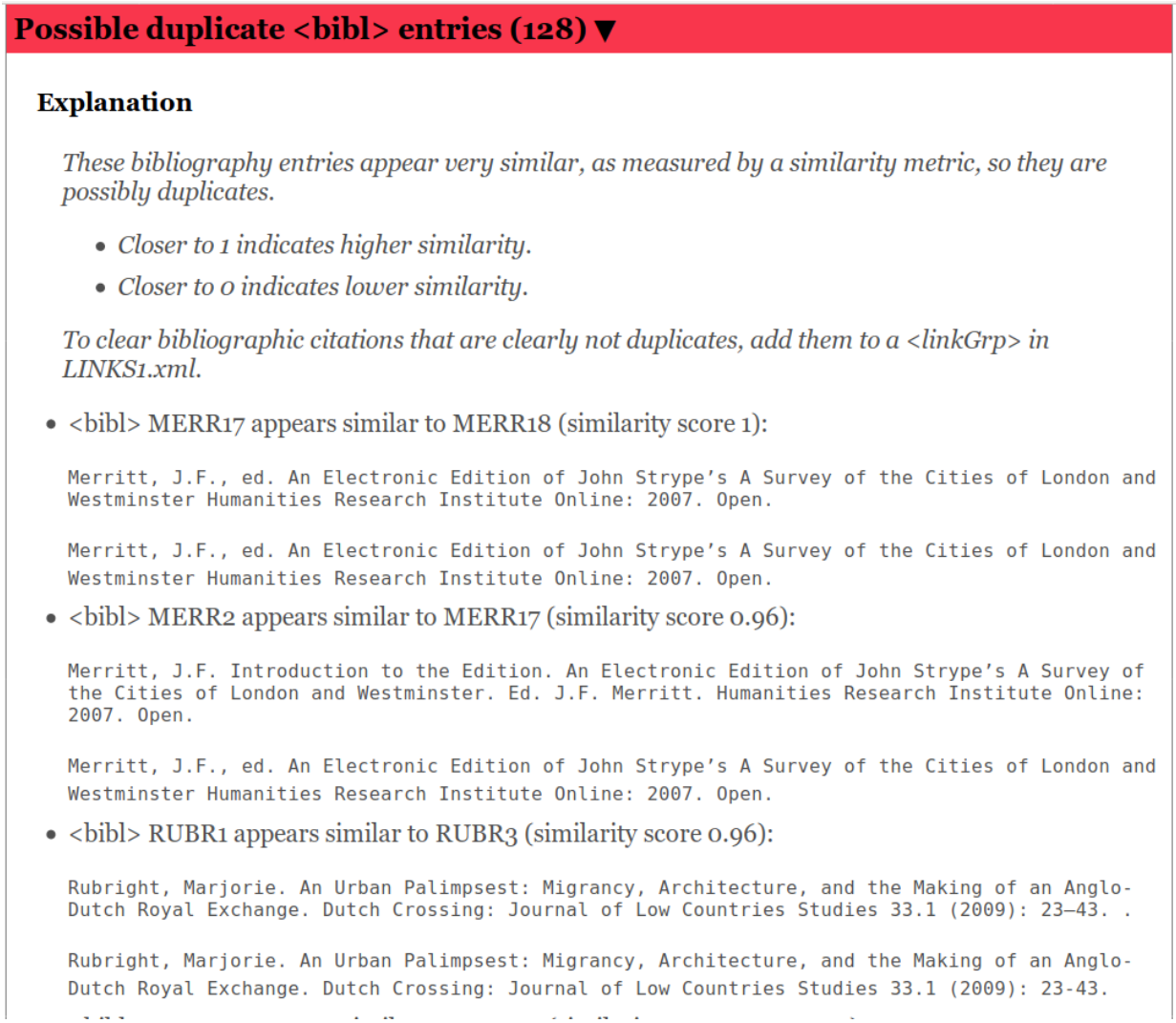


Figure 3: Results of a Level 2 diagnostic check that attempts to identify duplicate bibliography entries.

In the example above, the output informs the encoder that a bibliographic entry appears to be suspiciously similar to another. In each example, although both pairs of bibliographic entries *do* refer to the same work, they are substantively different as one citation references the edition itself while the other references the primary text. This is a common situation in projects that refer to both primary sources and their subsequent editions. For *MoEML*, this distinction must be retained, so the project can take further steps, such as adding more granular encoding of elements or adding pointers between the items to encode the relationship, to clear these results from their diagnostics output.

At Level 2, we also generate to-do lists for specific sub-projects, providing a set of tasks for the project team to focus on in order to reach a milestone or publish a particular document. The definition of "done" for a specific document extend beyond the document. For instance, before we deem a particular edition of a text publishable, we may require that all authority records (people, places, publications) linked from that document are themselves complete, so the to-do list for a given document may require work in a variety of other documents in the project. For example, Fig. 4 shows a list of referenced locations that need to be completed before *MoEML* can send a chapter of Stow's *Survey of London* out for peer-review.

Incomplete locations (17) ▼

These are locations that have not yet been published. Note that published does not denote a particular length of article. By published, we mean that we have done research into the location, attempted to map it on the Agas map and Google maps, and ensured it has some sort of abstract. It does not mean that an exhaustive account for the location has been written. All locations must be put into published before any large document can be sent to peer-review.

@xml:id	Name	Document status	Has abstract	Has agas	Has geo
THAM2	The Thames	assigned	✘	✔	✔
GUIL1	Guildhall	empty	✘	✔	✔
SUBR1	Suburb Without the Wall	empty	✘	✘	✘
TRIN2	Church of St. Trinity	empty	✘	✘	✘
MERC7	Merchant Taylors' Almshouses	empty	✘	✘	✘
MINO1	Minories	empty	✘	✔	✘
BISH2	Bishopsgate	empty	✘	✔	✔
HOGL3	Hog Lane (East Smithfield)	stub	✔	✔	✘
HOUN1	Houndsditch Street	stub	✔	✔	✔

Figure 4: A diagnostics to-do list to be completed before a chapter of *MoEML's* edition of Stow can be sent out for peer-review. *MoEML* requires that all locations contain a brief abstract, a zone outlined on the Agas map, and GIS coordinates (if possible); see Jenstad 2018 for more information.

Level 3

Armed with a comprehensive set of Level 1 and Level 2 diagnostics, and assuming our data is managed using a version-control repository such as Subversion or Git, we can now generate diachronic views of the project's progress. A script can check out a sequence of incarnations of the project, weekly over a period of months, for instance, and run the entire current diagnostic suite against it; we can then combine these snapshots to get a clear sense of how our work is proceeding. This also means that every time we develop a new diagnostic procedure, we can apply it to the entire history of the project to see the trajectory of project work with respect to the datapoint in question. Two examples, this time from the *Nxaʔamxcín Dictionary* project,[11] appear in Figs 5 and 6 below. Fig. 5 shows the number of completed dictionary entries in orange, rising steadily over a period of 18 months, and the number of occurrences of a known problem: duplicate instances of the same gloss. These duplicates rise along with the number of entries until October 2016, when this issue was added to our diagnostics process, and the encoders were able to address it.

11 The *Nxaʔamxcín Dictionary* is an indigenous dictionary project described in detail in Czaykowska-Higgins, Holmes, and Kell (2014).

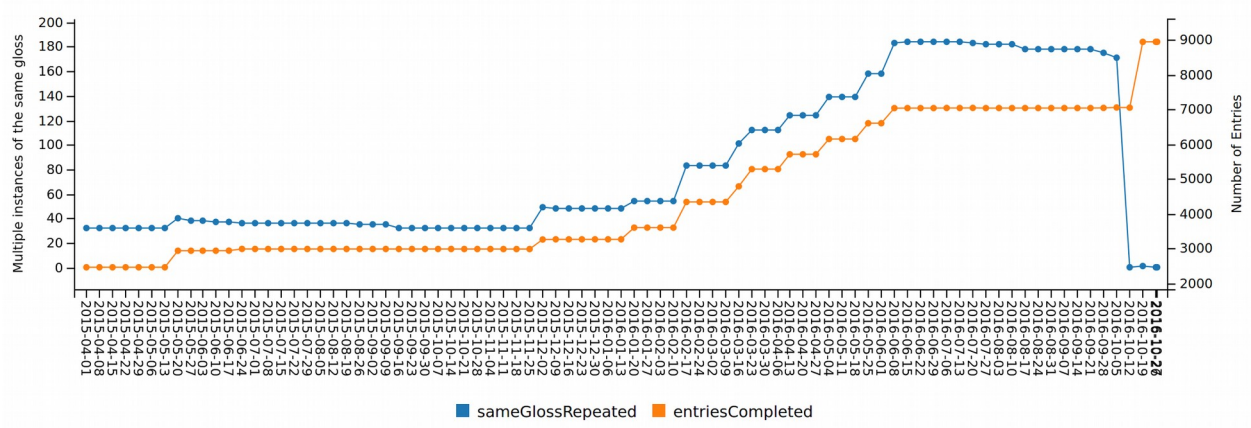


Figure 5: The number of instances of duplicate glosses, tracked against completed entries, in the *Nxaʔamxcín Dictionary* project.

Fig. 6 shows cases of broken cross-references, which also tend to increase along with the number of completed entries, but we can see from the graph that the issue was aggressively addressed in two separate campaigns in fall 2015 and summer 2016. New instances continue to appear, however.

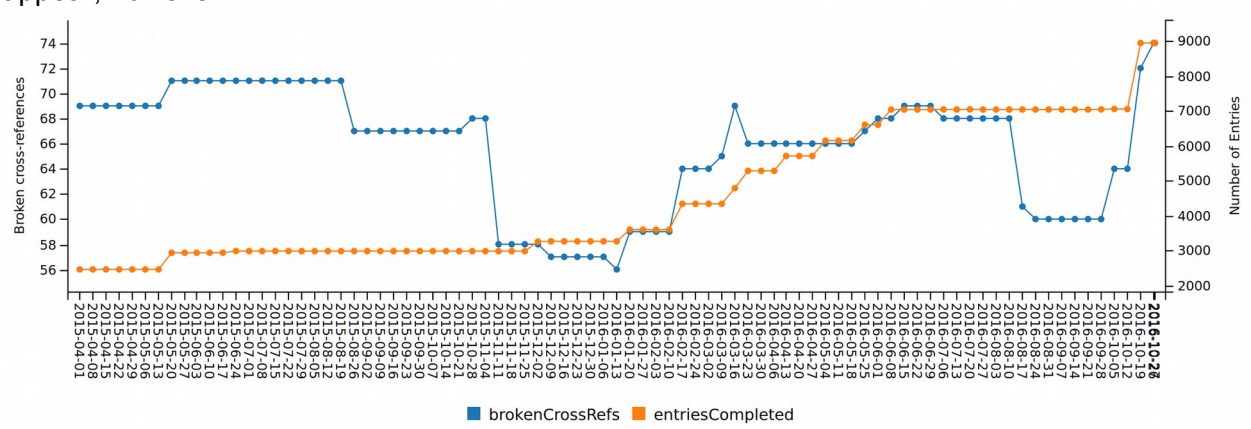
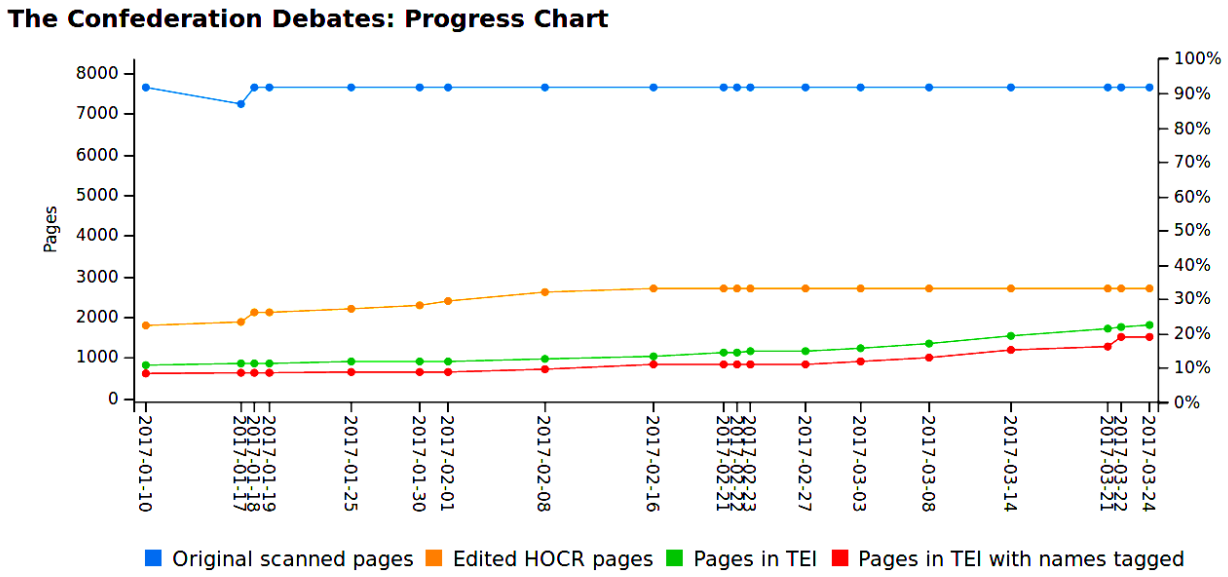


Figure 6: The number of broken cross-references, tracked against completed entries.

Fig. 7, from a different project, shows how this approach can be used to forecast completion dates for tasks in a project based on the progress rate so far.



Total names tagged so far: 5626

Problematic "unspecified" names: 129

Projected completion dates

- edited HOCR pages: 2017-08-04
- pages in TEI: 2017-11-16
- fully-name-tagged pages in TEI: 2018-01-15

Figure 7: Diachronic diagnostics used to project task completion dates.

Too much of a good thing

An automated build process which incorporates diagnostics can significantly aid in progress towards publication-readiness. However, it can also, paradoxically, slow a project down if not used with caution. Imagine a situation in which an encoder with the skills to add new diagnostic tests comes across an instance of an error which is not captured by the current set of tests. She might simply correct the error and move on; or she might write a new diagnostic to check for more instances of that error. The new diagnostic may uncover two thousand more examples, pushing back the project's completion date by several weeks. Diagnostics may also reveal new insights into the dataset which prompt changes in the working methods, or the encoding strategies. It may be more prudent to leave diagnostics for a specific target edition unchanged until that edition is actually published, and then refocus for the next edition.

Conclusion

As Matthew Kirschenbaum (2009) tells us, there “is no more satisfying sequence of characters” than “Done.” Despite the common tacit assumption that “nothing in digital form is ever really finished” (Burnard 2016), this is hardly a desirable state of affairs, and the ultimate purpose of a digital edition project is to finish and publish an edition, even if work continues on the project and further editions are planned. This requires not only that the document-level encoding be valid, but also that the entire dataset be coherent, consistent, and complete. To do so, digital edition projects must check their data both at the document level as well as the larger project level, ensuring that the project is coherent, consistent, and correct. Programmed diagnostics enable projects to enforce coherence and consistency, manage the workflow effectively, and measure their progress towards completeness.

References

Burnard, L. (2016). “How to Update your ODD.” TEI GitHub Repository. http://teic.github.io/PDF/purifyDoc.pdf.

Burnard, L. (2017). What is TEI Conformance, and Why Should You Care?. TEI 2017 Conference, Victoria, B.C. https://hcmc.uvic.ca/tei2017/abstracts/t_119_burnard_teiconformance.html.

Czaykowska-Higgins, E., Holmes, M., Kell, S. (2014). Using TEI for an Endangered Language Lexical Resource: The Nxaʔamxcín Database-Dictionary Project. *Language Documentation & Conservation* 8: 1–37.

Guidelines for Editors of Scholarly Editions (2016). *Modern Language Association*. https://www.mla.org/Resources/Research/Surveys-Reports-and-Other-Documents/Publishing-and-Scholarship/Reports-from-the-MLA-Committee-on-Scholarly-Editions/Guidelines-for-Editors-of-Scholarly-Editions.

Guiding Questions for Vetters of Scholarly Editions (2011). *Modern Language Association*. https://www.mla.org/content/download/3201/81158/cse_guidelines_2011.pdf.

Holmes, M. (2010). Using the Universal Similarity Metric to Map Correspondences between Witnesses. *Digital Humanities 2010 Conference Abstracts*, Kings College London. http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/html/ab-693.html.

Jacinto, M. H., Librelotto, G. R., José Carlos Ramalho, J. C., Henriques, P. R. (2002). Constraint specification languages : comparing XCSL, Schematron and XML-Schemas. http://repositorium.sdum.uminho.pt/handle/1822/619.

Jenstad, J. (2018). *The Map of Early Modern London*. University of Victoria. https://mapoflondon.uvic.ca.

Kirschenbaum, M. (2009). Done: Finishing Projects in the Digital Humanities. *DHQ* 3 (2). http://digitalhumanities.org:8081/dhq/vol/3/2/000037/000037.html.

Proceedings of the International Symposium on Quality Assurance and Quality Control in XML (2012). http://www.balisage.net/Proceedings/vol9/contents.html.

Rahtz, S. (2007). Technology Overview and Discussion: Data Capture, Editing, and Schemas. Oxford, February 13. http://tei.it.ox.ac.uk/Talks/2007-02-13-oucs/talk-editing.xml.

Sternbach, R., Okuda, M. (1991). *Star Trek, the next Generation: Technical Manual*. New York: Pocket Books. http://catalog.hathitrust.org/api/volumes/oclc/24648561.html.

Vorthmann, S., Robie, J. (2001). Beyond Schemas: Schema Adjuncts and the Outside World. *Markup Languages: Theory & Practice* 2 (3): 281–94.

Waldt, D. (2012). Quality Assurance in the XML World: Beyond Validation. http://www.balisage.net/Proceedings/vol9/author-pkg/Waldt01/BalisageVol9-Waldt01.html.